ENCAPSULATING PARAMETERIZED CELLS (PCELLS)

Inventors:

Jim E. Newton
Munich, Germany
Citizenship: U.S.A.

Christian Scheiba
Holzkirchen, Germany
Citizenship: Germany

Assignee:

Cadence Design Systems, Inc.
2655 Seely Avenue
San Jose, California 95134

Prepared By:
Jeffrey S. Smith
Bingham McCutchen LLP
Three Embarcadero Center, Suite 1800
San Francisco, California 94111
(650) 849-4422

ENCAPSULATING PARAMETERIZED CELLS (PCELLS)

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Application Number 60/394,108, filed July 2, 2002, which is hereby incorporated by reference in its entirety.

FIELD OF THE INVENTION

[0002] The present invention relates to the field of electronic circuit design and manufacturing.

BACKGROUND OF THE INVENTION

[0003] Circuit designers can create a design using a graphical tool to generate circuit components. In a parameterized cell (pcell), a designer selects parameters to describe features of electronic components for a design of an integrated circuit. The pcell tool can then automatically generate multiple representations of the electronic components of the pcell based on the parameters. The parameterized data specified by the user minimizes the effort of data entry and editing in the design tool, and also reduces design rule violations.

[0004] Any kind of layout data can be parameterized. For example, with a transistor pcell, the length, width, number of gate segments, and other design elements of the transistor, can be realized by simply inserting or changing one or more parameter values. For bipolar designs, parameterized data can include shapes such as arcs and circles. Design data can include text, and the location of the text may be relative to a virtual shape. Also, pieces of parameterized data can automatically appear, disappear, or replicate as a condition of another parameter.

[0005] To generate a pcell instance, a library name, cell name, instance name, orientation, and the type and values of parameters are specified. The pcell includes a formal parameter table where each row of the table contains a pcell parameter name, parameter type (such as float, integer, Boolean, or string), and a default value. The parameters for a resistor may include L, which specifies a length of a resistor segment; fingers, which specifies the number of fingers of the resistor; and q, which specifies a length of first and last connecting wires, for example.

[0006] The parameters may also be used to set a size of each parameterized cell shape (pshape). The parameters may be included in an expression having constants, formal parameters, and design rule identifiers. If a length or a width of a rectangle is not provided, a value may be found by searching through design rules. This may also be done to find values for a width of a path, a sub-path, or an arc. For example, a design rule may select the parameter for a poly width based on a design rule that determines minimum poly width.

[0007] A reference point is used to determine an initial location of a pcell. For example, a starting point of a pcell may be identified with specific coordinates, or a default starting point of (0, 0) may be used. The pshapes within a pcell can include alignment points such as, for example, vertices or a path centerline of a pshape, to align two pshapes in the pcell.

[0008] Traditional graphical tool systems draw the pcell according to its pcell parameters by executing code of a software program, such as SKILL code for example. The resulting pcell produced by the pcell code is considered as correct by definition. Moreover, if a parameter changes after this cell is created, the design system automatically triggers the recalculation of the shapes within the cell, and does so correctly (by definition). The pcell can then viewed from a higher level of the design hierarchy as a cell instance of the chip design.

[0009] However, the software program used to draw the pcell may have long or complicated algorithms which are inefficient. For example, if the pcell software program, when executed,

references other pcell parameters, design rules, and connectivity models, drawing the pcell may be time consuming. Therefore, there is a need for an efficient method of drawing pcells.

SUMMARY OF THE INVENTION

[00010] A method for encoding elements of an electronic design generates a flattened hierarchy of a parameterized cell of the electronic design, selects common and unique parameters of each element in the parameterized cell, and generates a physical design quantization characteristic value from the selected common and unique parameters.

BRIEF DESCRIPTION OF THE DRAWINGS

[00011] **Figure 1** shows an example of a method of creating a PDQ characteristic value for a pcell.

[00012] **Figure 2** shows an example of an array recognition method used during the performance of the PDQ creation method.

[00013] **Figures 3** through **7** show examples of arrays recognized during the performance of the array recognition method.

[00014] **Figure 8** shows an example of a system to automatically generate and verify a design for an electronic circuit using virtual shapes.

DETAILED DESCRIPTION

[00015] A graphical tool for designing an integrated circuit includes a method of generating a compressed representation of elements in a parameterized cell (pcell) of the design. The parameters common to instances of a shape in the pcell are stored in a shared field of a characteristic value data structure, and parameters unique to each instance of the shape are stored in individual fields. Also, shape instances that form an array within the pcell can be recognized, and an amount of information to represent them can be further reduced in the characteristic value data structure.

[00016] A Physical Design Quantization (PDQ) technique can be integrated into a pcell design tool to generate the characteristic value data structure, so that each pcell variant builds itself according to this fixed characteristic data structure that provides the PDQ Characteristic Value. Therefore, each pcell is drawn with reference to the PDQ characteristic value, instead of with reference to other pcell parameters, design rules, or connectivity models. This value is the same for every pcell view which is mask-wise identical, and contains enough information to completely regenerate the cell view. The characteristic value is relatively easy to parse, and uses a relatively small portion of the total memory storage of the design system.

Examples of the PDQ Characteristic Value

[00017] The examples of **Tables 1** and **2** depict characteristic values for two pcells, one small and one large. The data size of the characteristic value for the large case of **Table 2** is not much larger than that of the small case shown in **Table 1**. The data structure of **Table 1** shows the content of a PDQ Characteristic Value for a pcell master cell view containing two different geometries, a path geometry and a rectangle geometry. The data structure has a separate partition for each geometry. The path partition describes two paths, and the rectangle partition

describes ten rectangles. Each path entry contains a layer-purpose pair, a path width (W), a path style, a path starting point (x_1, y_1) and a path end point (x_2, y_2). The layer-purpose pair, width, and style are common parameters of each path shape, because these parameters are the same for each instance of the path shape.

[00018] The starting and end points are unique to each instance of the shape of the path. Each rectangle entry has common parameters including a layer-purpose pair, rectangle length and width (L, W); and uniqueness parameters for each instance of the rectangle shape, which, for this example, is the lower left-hand corner coordinate (x_{llc}, y_{llc}) of each instance. The ten rectangles of **Table 1** are represented with eight entries, because rectangle shape numbers 7 and 8 each contain lower left-hand coordinates for two instances of the same rectangle shape. (Similar structures are generated for polygons and labels as discussed below).

PDQ Characteristic Value for ("I1953" "t110d" "nflo_p" "layout")

data size for holding this PDQ Characteristic Value = 539
"partition for path geometry"

path shape no.	common parameters of the shape: layer	purpose	W	Style	unique para. of each shape inst.: x_1	y_1	x_2	y_2
1	("M0"	"drawing")	0.22	"truncateExtend"	((0.17	0.02)	((0.17	-4.69)))
2	("text"	"drawing1")	0.14	"truncateExtend"	((0.17	-0.1)	((0.17	-4.57)))

"partition for rectangle geometry"

rect. shape no.	common parameters of the shape: layer	purpose	L	W	unique param. of each shape instance: x_{llc}	y_{llc}	x_{llc}	y_{llc}
1	("text"	"drawing")	(0.05	0.05)	(0.03	-0.08))		
2	("instance"	"drawing")	(2.62	4.67)	(0.0	-4.67))		
3	("ND"	"drawing")	(2.05	4.67)	(0.33	-4.67))		
4	("ND"	"drawing")	(0.33	4.67)	(0.0	-4.67))		
5	("ND"	"drawing")	(0.24	4.67)	(2.38	-4.67))		
6	("GC"	"drawing")	(2.05	5.27)	(0.33	-4.97))		
7	("GC"	"drawing")	(2.05	0.2)	(0.33	-4.92)	(0.33	-4.67))
8	("CD"	"drawing")	(0.14	2.06)	(0.1	-4.57)	(0.24	-4.57))

TABLE 1

[00019] In most cases, a pcell master with a very large number of shapes can contain a high amount of regularity within it. For example, if a pcell contains multiple rectangles, and each rectangle has similar parameters, a PDQ Characteristic Value can describe the repetition of the

rectangle using an array partition, as shown in **Table 2**. Each array entry contains a layer-purpose pair; the array starting point, which, in this example, is the x and y coordinate of the lower left hand corner of the array (x_a, y_a); the x and y pitch within the array (dx, dy); the width and height of the rectangle which is arrayed (W, H), and the number of rows and columns contained in the rectangular array (row, col). Remaining shapes that are not described in the arrays may be described individually, as shown in the rectangle partition of **Table 2**.

PDQ Characteristic Value for ("I1976" "t110d" "nflo_p" "layout")
data size for holding this PDQ Characteristic Value = 549

"array partition"

array no.	layer	purpose	x_a	y_a	dx	dy	W	H	row	col
1	("text"	"drawing1")	0.1	-7.4	5.32	0.0	0.14	7.3	1	61)
2	("M0"	"drawing")	0.06	-7.52	5.32	0.0	0.22	7.54	1	61)
3	("GC"	"drawing")	0.33	-7.8	5.32	0.0	5.0	8.1	1	60)
4	("GC"	"drawing")	0.33	-7.75	5.32	7.8	5.0	0.2	2	60)
5	("CD"	"drawing")	0.1	-7.4	5.32	1.91	0.14	1.57	2	61)
6	("CD"	"drawing")	0.1	-3.57	5.32	1.91	0.14	1.56	2	61)

"rectangle partition"

rect.	common parameters				unique parameters			
shape no.	layer	purpose	L	W	x_{llc}	y_{llc}	x_{llc}	y_{llc}
1	("text"	"drawing")	(0.05	0.05)	(0.03	-0.08))		
2	("ND"	"drawing")	(318.88	7.5)	(0.33	-7.5))		
3	("ND"	"drawing")	(0.33	7.5)	(0.0	-7.5)	(319.21	-7.5))

TABLE 2

PDQ Methods

[00020] Methods to perform property creation, array recognition, and shape creation decoding are used to generate and process the PDQ data structure for a pcell.

[00021] An example of a method for encoding a PDQ characteristic value is shown in **Figure 1**. A flattened hierarchy of the pcell is copied to a temporary workspace, 110. The flattened hierarchy of the cell is compressed into a PDQ data structure by selecting common and unique parameters of each element in the pcell, 120. An array recognition process is applied to the PDQ data structure, 130.

[00022] In one embodiment, the PDQ structure is generated in a deterministic, sorted way, such that cell views which are geometrically equivalent produce the same data structure, regardless of the order that the shapes are stored in the cell view. This allows cell views which appear identical with respect to layers and shapes to produce the same PDQ characteristic value. For example, a PDQ data structure can be partitioned so that the partitions represent each type of geometry in the cell, such as labels, polygons, paths which do not look like rectangles, paths which look like rectangles, and rectangles, for example.

[00023] The compression process 120 in the PDQ encoding method differs for each geometry type. For example, for a given geometry, the shapes of the geometry are compressed into partitions that have common parameter information, or commonality criteria, describing each shape of the geometry, and unique parameter information, or uniqueness criteria, for the instances of each shape. Thus, the compression process for the pcell produces a partition for each type of geometry that includes the values common to each shape of the geometry listed once, and distinct values unique to each instance of the geometry shape.

[00024] An example of a geometry-type partition created by a compression method for a label has common parameters of a shape of the geometry, such as the layer/purpose, text, orientation, font, and height of the label for example, placed into a list. The origin of each label instance is a unique parameter of each instance.

[00025] A polygon's sorted geometry partition can have commonality criteria that contains the layer/purpose of a shape, and uniqueness criteria for each shape instance, such as the points of each instance's perimeter.

[00026] A partition for a non-rectangular path can have commonality criteria for a shape based on the shape's layer/purpose, width, and style, and the uniqueness criteria for each shape instance represented as a list of centerline points of the path.

[00027] A rectangle, or rectangular path, may have a partition including commonality criteria based on the layer/purpose, width and height of each rectangle. The uniqueness criteria for each rectangle instance of a shape may be the lower left corner of the instance. The partitions for the geometries are combined to form the PDQ characteristic value for the pcell.

[00028] An example of array recognition method 130 is shown in **Figure 2**, and may be used to further compress the amount of data that represents the shapes and instances of the pcell in the PDQ data structure. Each geometry may be partitioned into arrays if the number of instances exceeds a threshold value. This method examines a large list of instances of a shape, such a rectangle for example, and groups them into an array. For the rectangle, the array may be characterized by information for: the x and y coordinate of lower left hand corner of the array; the x (column) and y (row) pitch within the array; the x (width) and y (height) of the rectangle which is arrayed; and the number of rows and columns of rectangles contained in the array. The array recognition method is useful because an array of shapes can usually be drawn much faster than the individual shapes, the characterizing information can be stored in a much smaller amount of memory, and that smaller amount of storage space results in much faster parsing at decoding time.

[00029] As shown in **Figure 2**, a threshold value is selected to determine a minimum number of rectangles of an array, 210. The value of threshold may be 0, or any positive integer. For example, with a threshold of 0, arrays may be created that contain one or more shape instances. With a threshold of one, an array may be created if two or more instances of a shape are recognized.

[00030] In an encoded partition for rectangles, each instance of a rectangle shape shares commonality criteria, such as length and width, for example. Each instance of the shape may be uniquely identified by one of its boundary points, such as its lower left coordinate for example.

The list of commonality criteria and unique points may be sorted with a primary key such as x coordinate values, and secondary key, such as y coordinate values.

[00031] To recognize an array of rectangles, hash tables are created, 220. A primary key hash table is created to map each primary key, such as the x coordinate, to the list of points having that primary key. Each list of points is in increasing order by secondary key. A secondary key hash table is built to map each secondary key to the list of points having that secondary key. Each list of points in the secondary hash table is in increasing order by primary key.

[00032] The point with the minimum primary key value is determined, 230, which in this example is the minimum x coordinate, from the primary key hash table. If multiple points have the minimum primary key value, then a point with the minimum secondary key value is selected from this group of multiple points. This point can be determined by examining value of this x coordinate in the primary hash table. This point is the starting point of the array.

[00033] A set of delta values, such as delta x and delta y values for example, is determined, 240, from the primary and secondary hash tables. The set of possible values may be obtained by subtracting the x and y coordinates of the starting point with the x coordinates and y coordinates in the hash tables.

[00034] A maximum number of rows of rectangles is determined from the set, 250. For example, for the set of values of (delta_x delta_y rows cols) which make the following statements (1) through (4) true, choose the one which has the maximum value of rows * cols. If more than one have the same maximum value of rows * cols, then choose the one with the maximum value of rows.

$$0 < \text{rows} <= \text{number of distinct y values} \quad (1)$$

$$0 < \text{cols} <= \text{number of distinct x values} \quad (2)$$

The vector sum start_pt + c * (0 delta_y) + r * (delta_x 0) is a point

in the set, for all integers r and c, 0 <= r < rows AND 0 <= c < cols (3)

rows * cols > threshold (4)

[00035] If more than one such (delta_x delta_y rows cols) exists, with the same maximum product rows * cols, then one with the minimum delta_x is chosen. If that is still not unique, then the one of those with the minimum delta_y is chosen.

[00036] If no such (delta_x delta_y rows cols) exists, then the starting point is an individual rectangle, and no array is formed.

[00037] If (delta_x delta_y rows cols) exists, then an array is determined, 260, and the points from the primary and secondary hash tables, which correspond to points in this array, are removed from the hash tables. (This may remove a particular x coordinate or y coordinate from the hash tables).

[00038] The method returns to block 250 until no additional array is recognized.

Example of Generating a Characteristic Value with an Array Recognition Method

[00039] **Table 3** shows an example of a list of points that represent rectangle instances as shown in **Figure 3**. In this example, the rectangle instances have the same shape, which is a width of 0.4 and a height of 0.5, and are placed onto a 1 by 1 grid.

TABLE 3

(		(10 10)	(11 10)	(12 10)	(13 10)	(14 10)	(15 10)
			(11 11)	(12 11)	(13 11)		
			(11 12)	(12 12)	(13 12)	(14 12)	(15 12)
		(10 13)	(11 13)	(12 13)	(13 13)		
			(11 14)	(12 14)	(13 14)	(14 14)	(15 14)
)							

[00040] The array method identifies the following arrays:

[00041] Pass 1: (start_pt=(10 10) delta_x=1 delta_y=3 rows=2 cols=4).

[00042] Pass 2: (start_pt=(11 11) delta_x=1 delta_y=1 rows=2 cols=3).

[00043] Pass 3: (start_pt=(11 14) delta_x=1 delta_y=0 rows=1 cols=5).

[00044] Pass 4: (start_pt=(14 10) delta x=1 delta_y=2 rows=2 cols=2).

[00045] The first pass identifies (start_pt = (10 10) delta_x=1 delta_y=3 rows=2 cols=4), which is represented by the shaded rectangles shown in **Figure 4**. (There are other possible arrays to form with the same lower left hand rectangle, but none that group more than eight rectangles.) These shapes are eliminated before the next pass is done.

[00046] The second pass identifies the rectangular array structure (start_pt=(11 11) delta_x=1 delta_y=1 rows=2 cols=3), as shown with the shaded rectangle instances in **Figure 5**. There are two ways of grouping six rectangles starting with (11 11) : one with delta_y=1 and the other with delta_y=3. To solve ambiguous situations deterministically, the selected group is the one with the minimum delta_y.

[00047] The third pass identifies (start_pt=(11 14) delta_x=1 delta_y=0 rows=1 cols=5), as shown in **Figure 6**. These shape instances are eliminated before the next pass is done.

[00048] The fourth (and in this case final) pass identifies (start_pt=(14 10) delta_x=1 delta_y=2 rows=2 cols=2), as shown in **Figure 7**. The array partition for the pcell of **Figure 3** is then generated based on the arrays recognized in **Figures 4** through **7**.

Shape Drawing and Decoding Method

[00049] The method to create the pcell contents, given a PDQ Characteristic Value, uses the pcell design system's built-in functions to draw rectangles, paths, polygons, labels, and arrays of rectangles. The system decodes the parameters from the characteristic value to draw

the shapes of the pcell by using the shape parameters from the PDQ characteristic value as input values for the pcell drawing functions. These functions can include dbCreateRect, dbCreatePath, dbCreatePolygon, dbCreateLabel, and dbCreateViaShapeArray, for example, which are part of the Cadence pcell design tool. (If the pcell design system does not provide a draw-array function, it can be implemented programmatically with concurrent loops around the draw-rectangle function).

[00050] A design of an electronic circuit may include schematic symbols to provide a model for transistor-level simulation. With PDQ based pcells, the schematic design information, which is in electronic form, can be passed seamlessly through to the physical design tools as shown in **Figure 8**. Device parameters and connectivity data 820 are applied to symbols from symbol library 810 by the PDQ pcell device 830 to generate a schematic design of the electronic circuit. The PDQ pcell device 830 generates the components of the design using PDQ values with pcells. A cell placement and route device 840 accesses the pcells to edit the layout and to route the components of the design. A cell verification device then applies verification rules to the design to determine whether the design is functional 850.

[00051] The method of designing the electronic circuit using PDQ characteristic values for pcells may be performed by executing a computer software program. The software program may be stored in a computer readable medium, such as an electronic memory device for example. The program may be executed by a computer processing system, such as a micro-processing device, for example. The processing system may include devices found in a computing system, such as input/output devices, internal and external communication devices and buses, and power source devices, for example. The processing system may be implemented with hardware logic, or a combination of hardware and software.

[00052] These and other embodiments of the present invention may be realized in accordance with the above teachings and it should be evident that various modifications and changes may be made to the above described embodiments without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than restrictive sense and the invention measured only in terms of the claims.